Sep 76

Z80 Development System

Zilog

Product Specification

The Z-80 Development System is a turn-key unit designed to support all activities associated with the creation of microprocessor hardware and software. The system includes two floppy disks with a sophisticated file maintenance system. With this capability, the user can quickly retrieve, manipulate and store large files of data to minimize software development time. The system also includes an advanced real time debug module that connects directly to the user's system, thus providing a simultaneous hardware and software debug capability.

System Features

- Turn-key system including:
  - Z80-CPU with 4K bytes dedicated ROM memory
  - RS-232 or current-loop serial interface
  - 16K Bytes of read/write memory expandable to 60K bytes
  - Programmable hardware breakpoint module
  - Programmable real-time event storage module
  - In-circuit emulation bus to connect system to user's equipment
  - 2 floppy disk drives and controller
  - Full software including:
    - ROM based operating system
    - ROM based debug package
    - Editor
    - Assembler
    - File maintenance
  - Optional Universal Parallel I/O card for interface to printers, PROM programmers, etc.

System Description

The heart of the development system is the powerful Z-80 single chip microprocessor which is ideally suited to the multi-task operational requirements of a development system. A single Z80-CPU is shared between both the user's hardware (User Mode) and the System resident monitor (Monitor Mode).

In the Monitor Mode the System performs as a stand-alone development tool allowing software programs to be entered into RAM memory, edited, assembled, filed on disk for future use and loaded for execution. This entire process is quickly performed through simple commands from the user's terminal.

In the User Mode, the system memory and peripheral elements are dedicated to the user's own system. The system peripherals use I/O port numbers $E0_H$ through FF_H these port numbers are reserved for the system. In User Mode, a RAM resident user's program is executed in real time.

The use of RAM memory for the program eliminates costly and time-consuming PROM programming in the early phases of software development. The in-circuit emulation bus allows the user to connect his own peripheral devices or memory to the system and use them in conjunction with the system elements.

A major feature of the Z-80 is its powerful debug module. This module allows selected User Mode system transactions to be stored in real-time into a special memory. The user can also specify that various types of system transactions can suspend user operation and cause the system to reenter the Monitor Mode. A complete record of the 256 transactions that were recorded in the independent memory just prior to suspension can then be conveniently displayed on the system terminal or listed on a line printer. This ability to preserve real-time event sequences and then review selected events in detail, permits the user to accomplish product design and hardware/software debugging in the shortest time possible.

Hardware
Module Description

Processor Module

The Processor Module is a single card containing all elements necessary to function as a stand-alone computer. A serial asynchronous I/O port is provided for operation of a teletype or CRT terminal. The card also contains 3K bytes of ROM and 1K byte of RAM in which resides the operating system, peripheral drivers, bootstrap loader and debug software. The peripheral driver routines can be accessed by the user.

Real-Time Debug Module

The Zilog Development System real-time debugging capability enables the user to easily locate and correct any hardware or software design errors. With this module, the user monitors the operation of his software in real-time and sets hardware breakpoints to stop the program on any data, address or control bit pattern. Once stopped, the system returns to the Monitor Mode where the ROM resident debug software allows the user to display the contents of any internal CPU register, or memory location, or to change any register or memory location prior to continuing the program from that point.

The real-time debug module consists of a Real-Time Storage PCB and Breakpoint PCB.

The Real-Time Storage board contains a 256 x 32 storage array. This array stores up to 256 events. The 32 stored bits include: ■ 16-bit address bus, ■ 8-bit data bus ■ 7-bit control bus.

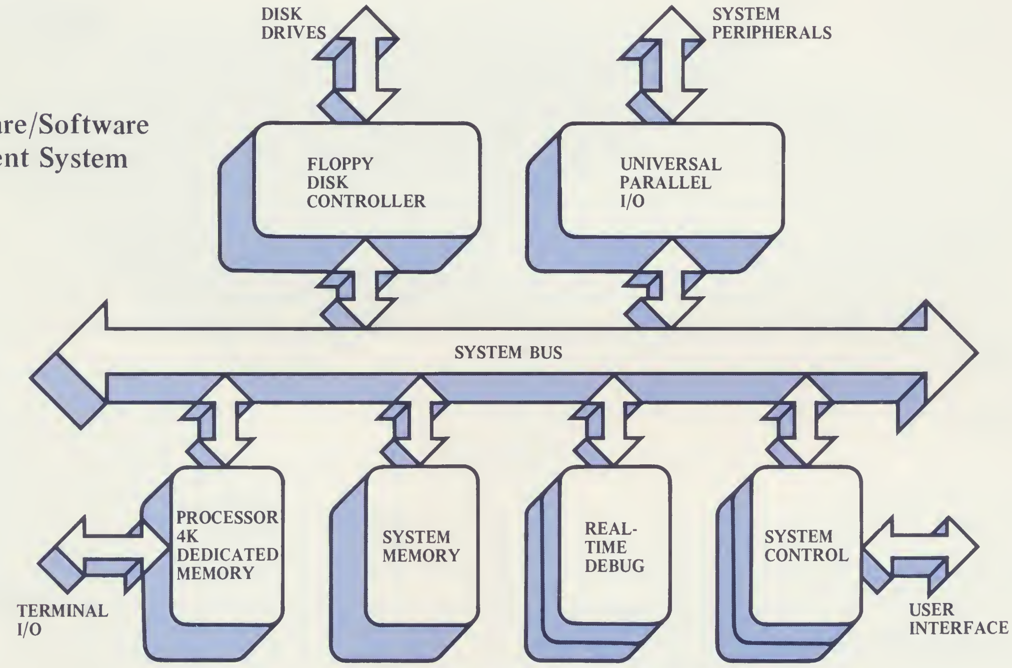
The last bit is used as a marker to identify the first transaction that is stored when the user's program begins execution. The debug software package allows the user to specify the type of transactions that are to be stored in the memory. Any combination of the following transactions can be stored:

■ Memory Reads ■ Memory Writes
■ I/O Port Reads ■ I/O Port Writes.

After the system returns to Monitor Mode from User Mode the contents of storage array can be printed on the user's terminal in a concise form so that he can analyze how he got to the current point in his program.

The Breakpoint card monitors the system bus and halts execution of a user's program if a user specified transaction occurs. The user may specify that a break should occur on any combination of the following transactions: ■ Memory Read ■ Memory Write ■ I/O Port Read ■ I/O Port Write. In addition he may specify that the selected transaction have: ■ A specified 16-bit address ■ Any specific bit pattern on the data bus. Thus the user can specify complex events such as writing a "1" on bit 6 of I/O port number $F6_H$.

Z80 Hardware/Software Development System



System Memory

The System uses standard, 4K dynamic RAM circuits configured in 4K byte increments up to a total of 60K bytes as required by the user. The standard system requires only 16K bytes but additional memory can be easily added if large programs are to be executed in User Mode. System memory is shared between the Monitor Mode and the User Mode.

In Monitor Mode programs are entered, edited, assembled and loaded directly into RAM memory for immediate execution without the additional cost and time delays associated with programming PROM's.

In the User Mode, RAM memory contains the user's software and the user has complete control over the system peripherals and CPU. The monitor does not use the first 60K byte memory locations while in User Mode. These locations are totally dedicated to the user. Any combination of external ROM, RAM or PROM can be used in place of, or in combination with the standard system RAM through the In-circuit Emulation Bus.

Floppy Disk Controller

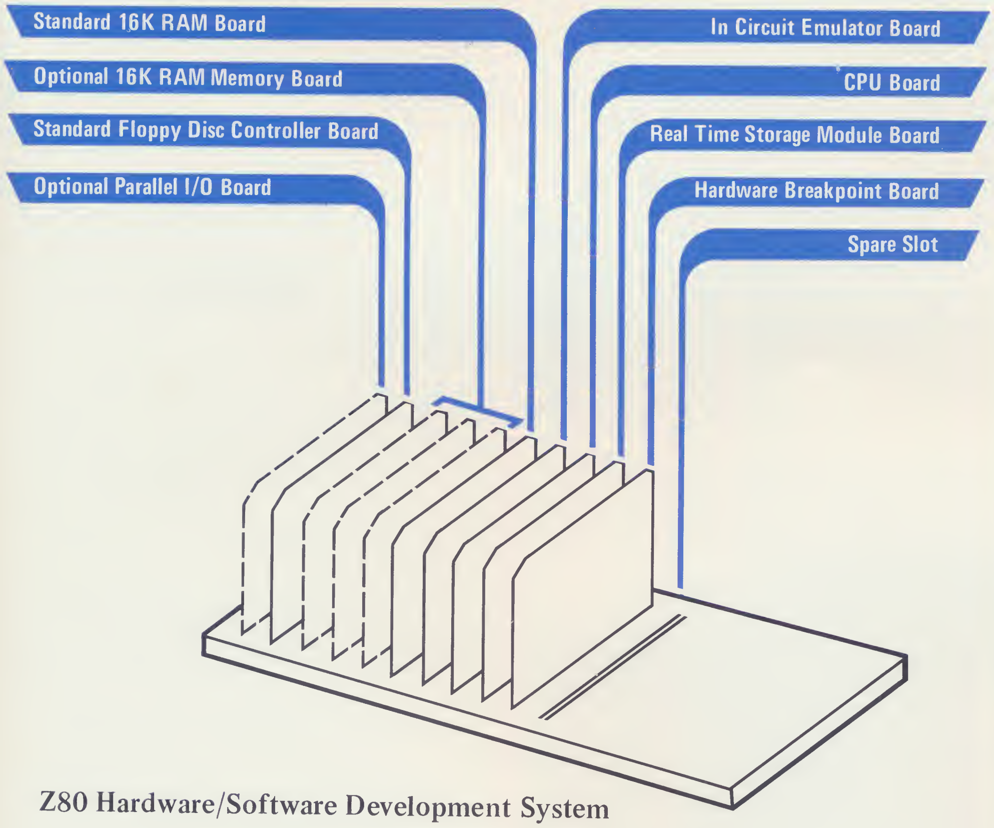
This single PCB interfaces two floppy disk drives in support of the Z-80 disk operating software. The ROM based monitor software contains the floppy disk software driver and bootstrap loader which loads the file maintenance software, system routines and user programs to be executed.

Universal Parallel I/O Module (Optional)

This module contains four Zilog parallel I/O controllers (Z80-PIO) which can control a wide range of parallel interface peripherals. The card is universal in that the system software can configure the Z80-PIO's with any set of bidirectional data transfer or any combination of status and control lines to match the requirements of various peripherals. It is used as an interface to optional peripherals such as line printers, paper tape punches and readers or PROM programmers.

In-Circuit Emulation Interface

This card contains all elements necessary to share the System between the User and Monitor Modes. In addition, a standard hardware interconnection cable is also provided for simple interface between the user's hardware and the System. This port includes: ■ 16-bit address bus ■ 8-bit data bus ■ All CPU control signals ■ System Clock ■ External Memory enable. All lines are fully buffered and provide TTL compatible signal levels for connection to any external user peripheral device, CPU control logic, memory system or even the user's own unique CPU card.



Z80 Hardware/Software Development System

Chassis Description

Spare slots in the system chassis are provided for additional user cards and I/O connectors. Seven slots are available; two for user cards; two for I/O connectors; and three for additional memory.

Software Description

The Z-80 Operating System

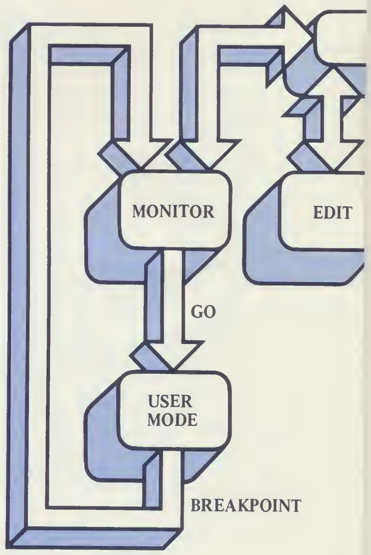
The Z-80 floppy disk based Development System is accessed through OS Z-80, which resides in one page (4K) of dedicated non-volatile memory. OS Z-80 controls the switching between User Mode and Monitor Mode, and provides a hierarchy of command levels. The monitor software also resides in the non-volatile 4K dedicated memory while the editor and assembler are stored on the floppy disk and are called into the general purpose system memory when required.

A cold start is initiated when power is turned on. The user then types "S" on his terminal and the system automatically determines the terminal's speed and adjusts itself accordingly. The terminal then responds by printing OS>. Four commands can be issued at this level: Debug, Edit, Assemble or File. When in the debug level, user programs can be executed with a simple GO command. The user's program will then continue to execute until a break is encountered.

Data and commands are entered into the system by the user in free-form format. That is, data and command fields are separated by any number of

F

INITI



Operating Systen

commas or spaces. In addition, commands may be fully spelled out or abbreviated. There are two special characters that can be used at any time:

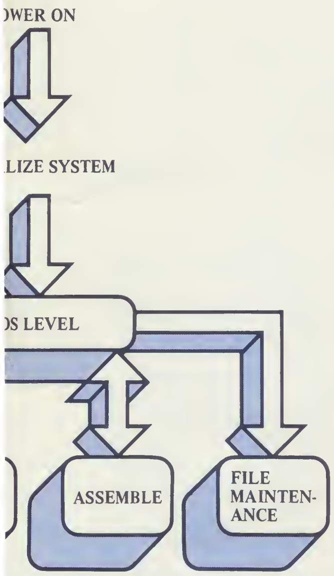
@ = delete last character
! = delete entire line

Debug Level

The Z80 Debug software has a repertoire of fourteen instructions, designed to give the user facilities in controlling, analyzing, and debugging his own programs which reside in up to 60K bytes of system memory. The debug commands:

BASE specifies the numerical base in which the user chooses to enter memory addresses and data. The base may be hexadecimal or decimal. When the base is unspecified, the system uses hexadecimal notation at this level.

BREAK sets an automatic hardware breakpoint into the real-time debug module. This break can be on a memory read, memory write, I/O port read, or I/O port write. Addresses, data and data masks can also be specified. A break from the user program can also be caused by pressing the Monitor button on the front panel. In either case, a break causes the state of the user's CPU to be stored so that execution can be resumed later Control returns to the debug level.



Flow of Control

COMPARE allows the user to compare two memory blocks of any size and list any locations that are not identical.

DISPLAY prints the contents of all CPU registers in a concise format. Alternately any individual register or any contiguous block of memory may be displayed.

GO begins execution of the user's program. Execution can begin at any specified address, or it can continue from a previous breakpoint. A hardware or manual break is required to return control back to the debug level.

HISTORY is normally issued after a break from a user program. This instruction lists on the terminal the state of the address, data and control busses of the CPU during the execution of up to 255 bus transactions that occurred in the users program just prior to a break.

LOAD transfers assembled programs into system memory, ready to be executed by the GO command.

MOVE allows the user to transfer a block of memory of any size from any location to any other location.

PULSE is identical to Break except that a pulse on a special connector is provided each time the specified condition occurs and the program continues to execute. Pulse can be used to synchronize an oscilloscope display.

SAVE stores the RAM image of linked programs and subroutines on the user's disk.

SET stores data entered from the terminal into specified registers or memory locations.

STEP executes one instruction and then prints the contents of the CPU registers.

TRACE specifies if memory read, memory write, port read and/or port write conditions are to be stored in the Real-Time Debug Module during execution of the user's program.

QUIT returns control to the OS level.

Editor Level

The Z-80 text editor is called from the OS level by the command:

EDIT filename filetype

The EDITOR transfers the user's file from disk to a work space in memory. The EDITOR works on a pointer concept where a pointer is moved by the user to access any desired line. The user can modify this file by any of the following commands:

AGAIN repeats the previous command

BOTTOM moves the line pointer to the bottom line.

CHANGE locates any specified character string and replaces it with any new character string as entered from the terminal. The user can specify how many occurrences per line as well as the maximum number of lines to change.

DELETE removes a specified number of lines from the file.

FILE writes the file from the work space on to the user's disk and returns to the OS level.

GET loads a block of text temporarily stored on the disk back into the work space to the user's disk and returns

GOTO places the pointer at a specified line number.

INSERT transfers new text from the terminal to the location of the pointer in the work space.

LINENO prints out the line number of the current pointer position.

LOCATE locates any specified string in the text and places the pointer at the beginning of the first line in which it occurs.

MACRO concatenates a string of editor commands, coupled with an & symbol. MACRO is executed with the XECUTE command.

NEXT moves the pointer down by a specified number of lines in the work space.

PRINT outputs the specified number of lines to the terminal beginning from the current pointer position.

PUT writes a specified block of text on the disk for temporary storage. The text is recovered with the GET command.

QUIT returns control to the OS level without filing the text on the disk.

REPLACE replaces the current line of text with new text from the terminal.

SAVE files the contents of the work space on the disk and returns to the EDITOR.

TOP moves the line pointer to the zero position in front of the first line.

(continued)

UP moves the pointer up a specified number of lines in the work space.

XECUTE executes the Macro command.

Assembler Level

The Z-80 resident assembler is a counterpart of the Z-80 cross-assembler and processes the same source programs.

The assembler is entered from the OS level by the command

ASSEMBLE filename filetype

The source program is written in assembly language. Free-form format of the instructions is permitted within the following rules:

label: OP code operand-1 operand-2; comment

Any number of commas or blanks may be used to delimit the terms. Some instructions possess no or only one operand. Labels are terminated by a colon unless they begin in column 1, in which case the colon is optional. OP codes cannot begin in column 1. Comments begin with a semicolon and can begin in any column.

File Maintenance Level

The file maintenance system has a repertoire of twelve commands that allow the user to easily manipulate large files of any type. These commands include:

APPEND one file to another file.

COMBINE creates a new file from any number of existing files.

COPY copies the entire contents of one diskette to another diskette

COMPACT removes any unused information in the directory.

DUMP performs a hexidecimal dump of a file to the terminal.

ERASE eliminates a file from the disk.

FORMAT initializes a diskette for system operation

LIST prints the directory of all files on the user's disk.

NAME changes the name associated with a file.

PRINT lists a file on the terminal.

QUIT causes the system to re-enter OS.

STAT gives the number of sectors that are unused on the disk.

Logical Device Assignments

All system I/O activity is oriented to six logical devices with the following default assignments:

Logical Device	Default Assignment
1) Console Input	Console terminal
2) Console Output	Console terminal
3) System Input	Floppy disk
4) System Output	Floppy disk
5) Utility Input	Console terminal
6) Utility Output	Console terminal

Logical devices 1 and 2 are used to receive console commands and communicate their direct results.

Logical devices 3 and 4 are used to input user files and for output of system operations such as editing and assembling.

Logical devices 5 and 6 are used for special I/O activity such as reading and punching paper tapes, and printing user files.

Default assignments can be overridden by the command:

ASSIGN device parm

Where "device" is one of the following six logical device abbreviations

1) CONIN
2) CONOUT
3) SYSIN
4) SYSOUT
5) UTLIN
6) UTLOUT

and "parm" is either TTY or ZDOS for the standard device assignment of the console terminal or floppy disk, respectively, or the actual address of a special I/O routine for manipulating a custom I/O device.

This command can be used to interface I/O devices not supported on the Z-80 Development System.

For example,

ASSIGN CONOUT 1740

will direct all console output to address 1740. By loading the I/O handler for a special display device at this address, the results of user commands will appear on this device.

Dual Floppy Disc Subsystem

Specifications

CPU
Standard Z-80 CPU

Memory
3K bytes ROM/1K bytes static RAM dedicated to system monitor

16 bytes general purpose RAM expandable to 60K bytes

System Clock
Crystal controlled at 2.0 MHz or optional 2.5 MHz

I/O Channels
Standard Interface to Disk Unit, Serial Data Terminal and In-Circuit Emulator. Optional Interface for PROM Programmer and Line Printer. Two spare connectors for other user designated system peripherals.

Standard Equipment
Z-80 CPU Card with 4K bytes of ROM/RAM Monitor and debug Software

16K Bytes of RAM

Realtime Debug Module (Storage module and breakpoint)

Dual Floppy Disk Subsystem

In-Circuit Emulator

RS-232 or Current Loop Asynchronous Terminal Interface

Z-80 Resident Assembler, Editor, Disk Operating System and File Maintenance System

Full Documentation

Optional Modules
8 port parallel I/O Modules with Interrupt

RAM, 16K byte Memory Modules

Module Extender

Drawer Slides and Extenders for Rack Mounting

AC Power Requirement
50/60 Hz, 115 VAC, 200 Watts

Optional 230 VAC Power Supply

Environmental Characteristics
Operating Temperature: 0° to 50°C

Physical Characteristics
Two separate chassis. One contains the disk drives and disk power supplies, while the other contains all other elements

Approximate weights and dimensions apply to both chassis:
Size: 19"W x 9"H x 15"D Weight: 35 lbs.

Electrical
Integral Power Supplies provide all necessary voltages, plus 4 amps of +5V ±5% is available in the CPU chassis for user cards

Zilog 170 State Street / Los Altos, California 94022 / (415) 941-5055 TWX 910-370-7955

Printed in U.S.A.

Sep 76

Z80-CPU Product Specification

Zilog

The Zilog Z80 product line is a complete set of microcomputer components, development systems and support software. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

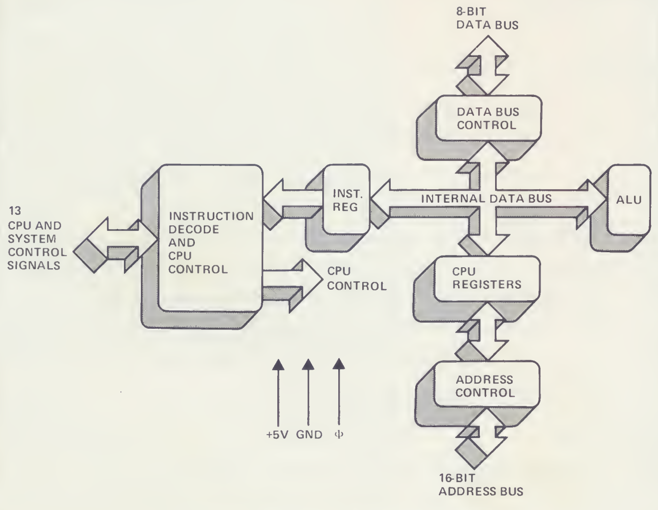
The Z80-CPU is a third generation single chip microprocessor with unrivaled computational power. This increased computational power results in higher system through-put and more efficient memory utilization when compared to second generation microcomputers. In addition, the Z80-CPU is very easy to implement into a system because of its single voltage requirement plus all output signals are fully decoded and timed to control standard memory or peripheral circuits. The circuit is implemented using an N-channel, ion implanted, silicon gate MOS process.

Figure 1 is a block diagram of the CPU, Figure 2 details the internal register configuration which contains 208 bits of Read/Write memory that are accessible to the programmer. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or as 16-bit register pairs. There are also two sets of accumulator and flag registers. The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast Interrupt response. The CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

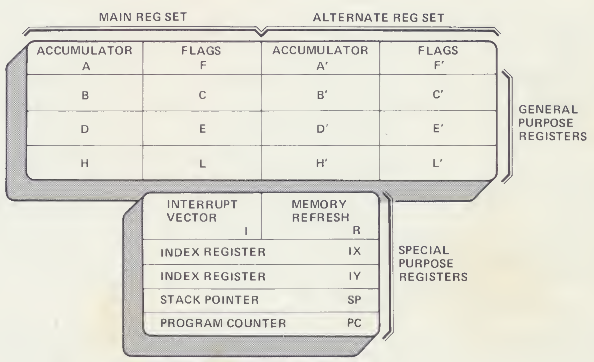
The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The Refresh register provides for automatic, totally transparent refresh of external dynamic memories. The I register is used in a powerful interrupt response mode to form the upper 8 bits of a pointer to a interrupt service address table, while the interrupting device supplies the lower 8 bits of the pointer. A call is then made to this service address.

FEATURES

- Single chip, N-channel Silicon Gate CPU.
- 158 instructions—includes all 78 of the 8080A instructions with total software compatibility. New instructions include 4-, 8- and 16-bit operations with more useful addressing modes such as indexed, bit and relative.
- 17 internal registers.
- Three modes of fast interrupt response plus a non-maskable interrupt.
- Directly interfaces standard speed static or dynamic memories with virtually no external logic.
- 1.6 μs instruction execution speed.
- Single 5 VDC supply and single-phase 5 volt Clock.
- Out-performs any other single chip microcomputer in 4-, 8-, or 16-bit applications.
- All pins TTL Compatible
- Built-in dynamic RAM refresh circuitry.

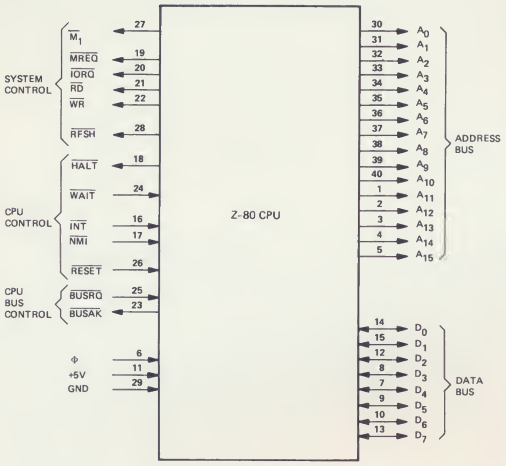


Z80-CPU BLOCK DIAGRAM



Z80-CPU REGISTERS

Z80-CPU Pin Description



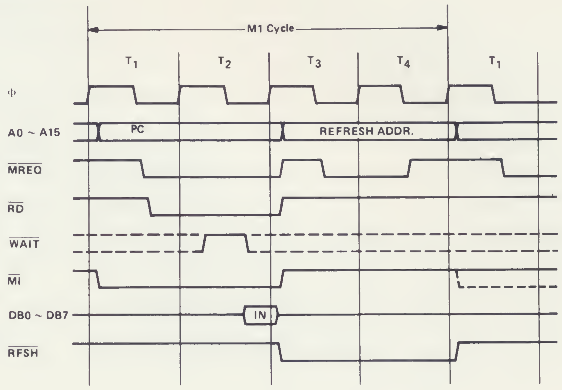
Z80-CPU PIN CONFIGURATION

A_0-A_{15} (Address Bus)	Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges.
D_0-D_7 (Data Bus)	Tri-state input/output, active high. D_0 - D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.
$\overline{M_1}$ (Machine Cycle one)	Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution.
$\overline{MREQ}$ (Memory Request)	Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.
$\overline{IORQ}$ (Input/ Output Request)	Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus.
$\overline{RD}$ (Memory Read)	Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
$\overline{WR}$ (Memory Write)	Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
$\overline{RFSH}$ (Refresh)	Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.
$\overline{HALT}$ (Halt state)	Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.
$\overline{WAIT}$ (Wait)	Input, active low. $\overline{WAIT}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active.
$\overline{INT}$ (Interrupt Request)	Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{BUSRQ}$ signal is not active.
$\overline{NMI}$ (Non Maskable Interrupt)	Input, active low. The non-maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H.
$\overline{RESET}$	Input, active low. $\overline{RESET}$ initializes the CPU as follows: reset interrupt enable flip-flop, clear PC and registers I and R and set interrupt to 8080A mode. During reset time, the address and data bus go to a high impedance state and all control output signals go to the inactive state.
$\overline{BUSRQ}$ (Bus Request)	Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these busses.
$\overline{BUSAK}$ (Bus Acknowledge)	Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Timing Waveforms

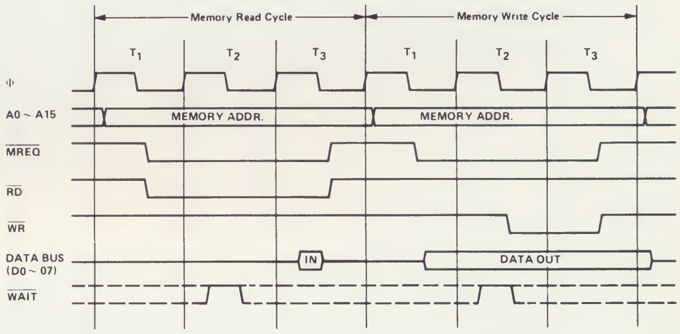
INSTRUCTION OP CODE FETCH

The program counter content (PC) is placed on the address bus immediately at the start of the cycle. One half clock time later MREQ goes active. The falling edge of MREQ can be used directly as a chip enable to dynamic memories. RD when active indicates that the memory data should be enabled onto the CPU data bus. The CPU samples data with the rising edge of the clock state T_3. Clock states T_3 and T_4 of a fetch cycle are used to refresh dynamic memories while the CPU is internally decoding and executing the instruction. The refresh control signal RFSH indicates that a refresh read of all dynamic memories should be accomplished.



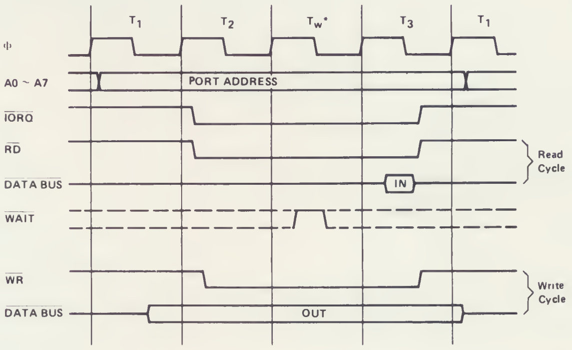
MEMORY READ OR WRITE CYCLES

Illustrated here is the timing of memory read or write cycles other than an OP code fetch (M_1 cycle). The MREQ and RD signals are used exactly as in the fetch cycle. In the case of a memory write cycle, the MREQ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory.



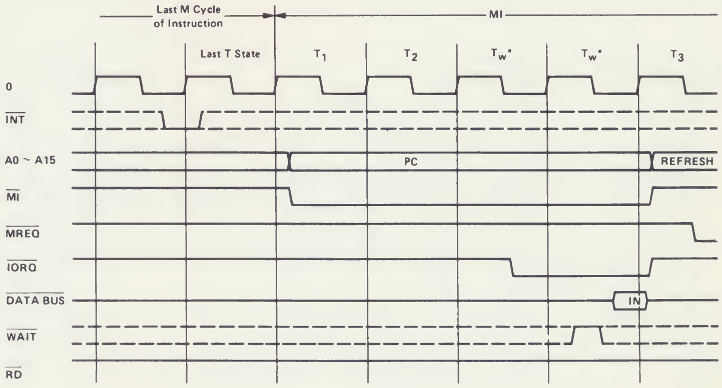
INPUT OR OUTPUT CYCLES

Illustrated here is the timing for an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted (Tw*). The reason for this is during I/O operations this extra state allows sufficient time for an I/O port to decode its address and activate the WAIT line if a wait is required.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

The interrupt signal is sampled by the CPU with the rising edge of the last clock at the end of any instruction. When an interrupt is accepted, a special M_1 cycle is generated. During this M_1 cycle, the IORQ signal becomes active (instead of MREQ) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states (Tw*) are automatically added to this cycle so that a ripple priority interrupt scheme, such as the one used in the Z80 peripheral controllers, can be easily implemented.



Z80 Instruction Set

The following is a summary of the Z80 instruction set showing the assembly language mnemonic and the symbolic operation performed by the instruction. A more detailed listing appears in the Z80-CPU technical manual. The instructions are divided into the following categories:

- 8-bit loads
- 16-bit loads
- Exchanges
- Memory Block Moves
- Memory Block Searches
- 8-bit arithmetic and logic
- 16-bit arithmetic
- General purpose Accumulator & Flag Operations
- Miscellaneous Group
- Rotates and Shifts
- Bit Set, Reset and Test
- Input and Output
- Jumps
- Calls
- Restarts
- Returns

In the table the following terminology is used.

b ≡ a bit number in any 8-bit register or memory location
cc ≡ flag condition code
- NZ ≡ non zero
- Z ≡ zero
- NC ≡ non carry
- C ≡ carry
- PO ≡ Parity odd or no over flow
- PE ≡ Parity even or over flow
- P ≡ Positive
- M ≡ Negative (minus)

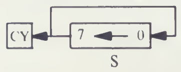
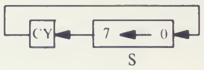
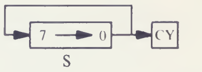
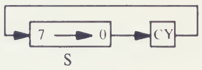
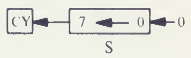
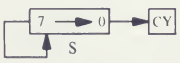
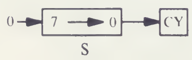
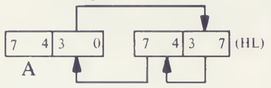
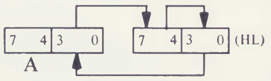
d ≡ any 8-bit destination register or memory location
dd ≡ any 16-bit destination register or memory location
e ≡ 8-bit signed 2's complement displacement used in relative jumps and indexed addressing
L ≡ 8 special call locations in page zero. In decimal notation these are 0, 8, 16, 24, 32, 40, 48 and 56
n ≡ any 8-bit binary number
nn ≡ any 16-bit binary number
r ≡ any 8-bit general purpose register (A, B, C, D, E, H, or L)
s ≡ any 8-bit source register or memory location
s_b ≡ a bit in a specific 8-bit register or memory location
ss ≡ any 16-bit source register or memory location
subscript "L" ≡ the low order 8 bits of a 16-bit register
subscript "H" ≡ the high order 8 bits of a 16-bit register
( ) ≡ the contents within the ( ) are to be used as a pointer to a memory location or I/O port number

8-bit registers are A, B, C, D, E, H, L, I and R
16-bit register pairs are AF, BC, DE and HL
16-bit registers are SP, PC, IX and IY

Addressing Modes implemented include combinations of the following: Immediate, Immediate extended, Modified Page Zero, Relative, Extended, Indexed, Register, Implied, Register Indirect, Bit

	Mnemonic	Symbolic Operation	Comments
8-BIT LOADS	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16-BIT LOADS	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H; (SP-2) \leftarrow ss_L$	ss = BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP); dd_H \leftarrow (SP+1)$	dd = BC, DE, HL, AF, IX, IY
EXCHANGES	EX DE, HL	DE ↔ HL	
	EX AF, AF′	AF ↔ AF′	
	EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	
	EX (SP), ss	$(SP) \leftrightarrow ss_L, (SP+1) \leftrightarrow ss_H$	ss ≡ HL, IX, IY

	Mnemonic	Symbolic Operation	Comments
MEMORY BLOCK MOVES	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
MEMORY BLOCK SEARCHES	CPI	A-(HL), HL ← HL+1 BC ← BC-1	
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	A-(HL) sets the flags only. A is not affected
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, Repeat until BC= 0 or A = (HL)	
8-BIT ALU	ADD s	A ← A + s	
	ADC s	A ← A + s + CY	CY is the carry flag
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	$A \leftarrow A \wedge s$	
	OR s	$A \leftarrow A \vee s$	
	XOR s	$A \leftarrow A \oplus s$	

Group	Mnemonic	Symbolic Operation	Comments
8-BIT ALU	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX±e), (IY+e)
	DEC d	d ← d - 1	
16-BIT ARITHMETIC	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
GP ACC. & FLAG	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	$A \leftarrow \overline{A} + 1$	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
MISCELLANEOUS	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
ROTATES AND SHIFTS	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

Group	Mnemonic	Symbolic Operation	Comments
BIT S, R, & T	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag
	SET b, s	$s_b \leftarrow 1$	s ≡ r, (HL) (IX+e), (IY+e)
	RES b, s	$s_b \leftarrow 0$	
INPUT AND OUTPUT	IN A (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
JUMPS	JP nn	PC ← nn	cc { NZ PO, Z PE, NC P, C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	
	JR kk, e	If condition kk is true PC ← PC + e, else continue	kk { NZ NC, Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
CALLS	CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, PC ← nn	cc { NZ PO, Z PE, NC P, C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
RESTARTS	RST L	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$ $PC_L \leftarrow L$	
RETURNS	RET	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$	
	RET cc	If condition cc is false continue, else same as RET	
	RETI	Return from interrupt, same as RET	
	RETN	Return from non- maskable interrupt	

A.C. Characteristics

$T_A = 0°C$ to $70°C$, $V_{cc} = +5V \pm 5\%$, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min.	Max.	Unit	Test Condion
Φ	t_c	Clock Period	.4	2	µsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180		nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180		nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		200	nsec	C_L=100pF
	$t_{F(AD)}$	Delay to Float		100	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{RD}$ or $\overline{WR}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		350	nsec	C_L=100pF
	$t_{F(D)}$	Delay to Float During Write Cycle		100	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	100		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	100		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		130	nsec	C_L=50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		130	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		150	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		110	nsec	C_L=50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		130	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		130	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		150	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		130	nsec	C_L=50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		150	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		130	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		150	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		110	nsec	C_L=50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		130	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge at Clock, $\overline{WR}$ High		150	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		160	nsec	C_L=30pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		220	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		200	nsec	C_L=30pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		200	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		240	nsec	C_L=50pF
$\overline{INT}$	$t_{s\,IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	70		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	60		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	70		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		150	nsec	C_L=50pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		150	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	70		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 120$

[2] $t_{aci} = t_c - 140$

[3] $t_{ca} = t_{w(\Phi H)} + t_f - 80$

[4] $t_{caf} = t_{w(\Phi H)} + f - 100$

[5] $t_{dcm} = t_c - 300$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 330$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

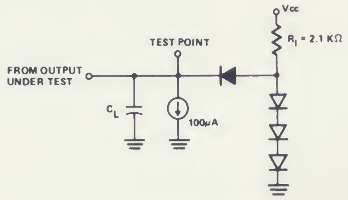
[8] $t_{w(\overline{MRH})} = t_c - 80$

[9] $t_{w(\overline{MRL})} = t_{w(\Phi H)} + t_f - 70$

[10] $t_{w(\overline{WR})} = t_c - 80$

NOTES:

1. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.
2. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
3. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

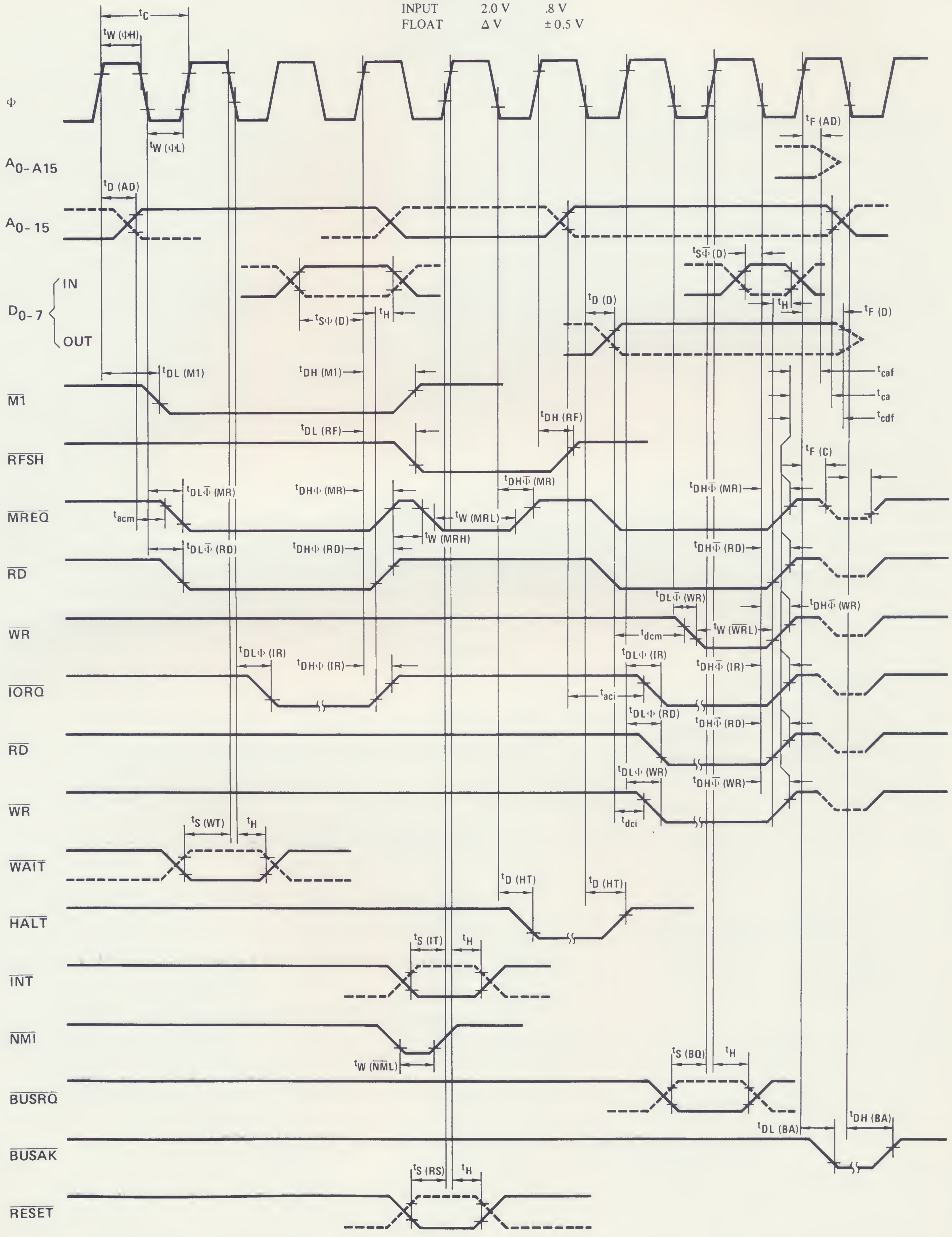


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2 V	.8 V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	Δ V	± 0.5 V



Absolute Maximum Ratings

Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. Characteristics

$T_A = 0°C$ to $70°C$, $V_{cc} = 5V \pm 5\%$ unless otherwise specified

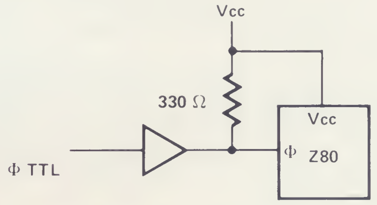
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} [1]		V_{cc}	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH}= -100μA
I_{CC}	Power Supply Current			200	mA	t_c=400nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

$T_A = 25°C$, f = 1 MHz,
unmeasured pins returned to ground

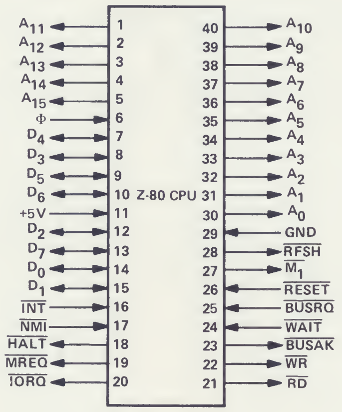
Symbol	Parameter	Max.	Unit
C_Φ	Clock Capacitance	20	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

[1] Clock Driver

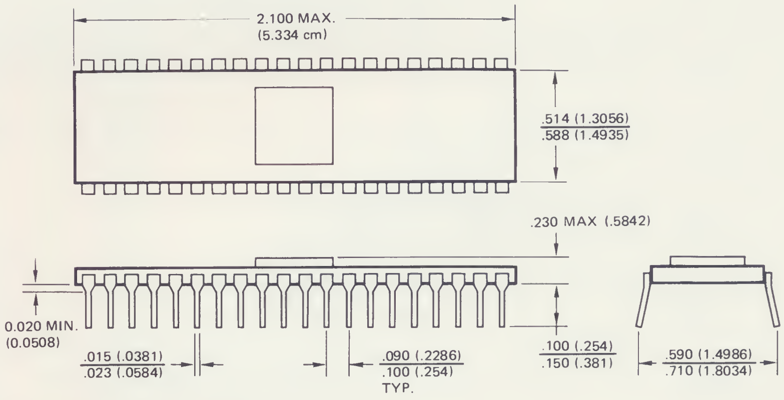


An external clock pull-up resistor of (330Ω) will meet both the A.C. and D.C. clock requirements.

Package Configuration



Package Outline



*Dimensions for metric system are in parenthesis

Zilog 170 State Street / Los Altos, California 94022 / (415) 941-5055 TWX 910-370-7955

Printed in U.S.A.

4 76.2 10K